



# ASSIGNMENT 1

ITRI 626

ENRICO DREYER  
31210783

## Table of Contents

Introduction .....	2
What I did.....	2
Outcome .....	5
References .....	6

## Table Of Figures

Figure 1: Getting the derivatives .....	2
Figure 2: Plot of a quadratic function (Mesquita, 2021).....	3
Figure 3: Getting repetitions of predictions.....	4
Figure 4: Drawings on board .....	5
Figure 5: Final outcome .....	6

## Introduction

In this assignment I learned about simplest form of learning, which is hot and cold learning. After making a prediction, you predict again with one higher weight and one lower weight. You then move to the one with the lowest error. This will result in an error that is close to 0.

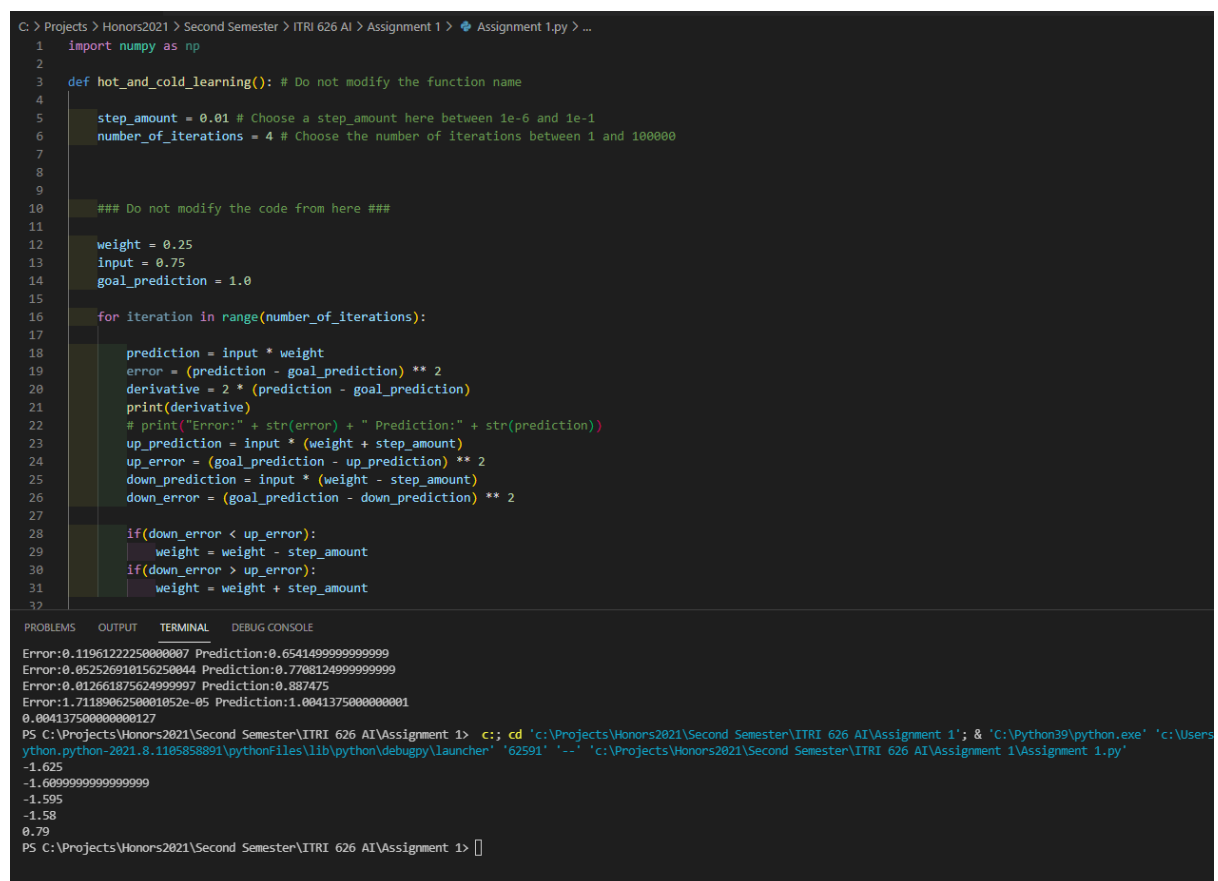
A snippet of code was given to us, where we had to determine the `step_amount` and `number_of_iterations` to predict a value that is closest to the `goal_prediction` that is 1.

## What I did

I googled Hyperparameter Optimization as suggested in class. This helped me understand what the assignment was about. I also found the website <https://realpython.com/python-ai-neural-network/>, this really helped me grasp on what to do in the assignment.

I read that calculating the derivative ( $2 * \text{prediction} - \text{goal\_prediction}$ ) you get to see if your `step_amount` should increase or decrease. As shown in the screenshot bellow, I started with a `step_amount` of 0.01 and the `number_of_iterations` of 4.

I also commented out the `print("Error:" + str(error) + " Prediction:" + str(prediction))` to make it easier for me to get the derivative.

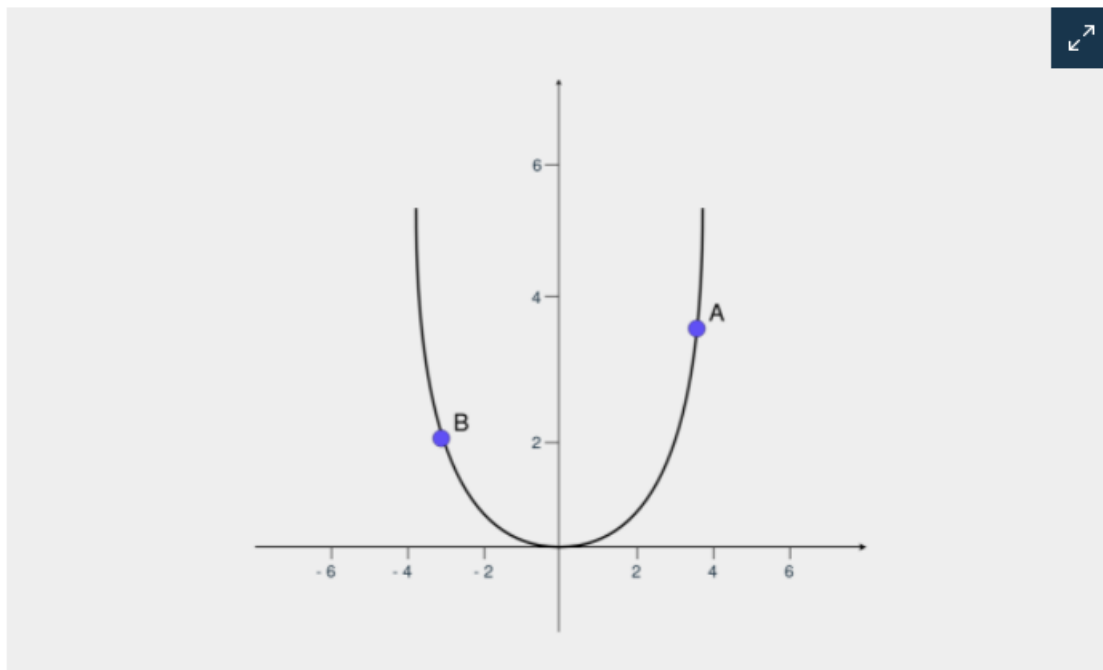


```
C:\> Projects > Honors2021 > Second Semester > ITRI 626 AI > Assignment 1 > Assignment 1.py > ...
1  import numpy as np
2
3  def hot_and_cold_learning(): # Do not modify the function name
4
5      step_amount = 0.01 # Choose a step_amount here between 1e-6 and 1e-1
6      number_of_iterations = 4 # Choose the number of iterations between 1 and 100000
7
8
9
10     ### Do not modify the code from here ###
11
12     weight = 0.25
13     input = 0.75
14     goal_prediction = 1.0
15
16     for iteration in range(number_of_iterations):
17
18         prediction = input * weight
19         error = (prediction - goal_prediction) ** 2
20         derivative = 2 * (prediction - goal_prediction)
21         print(derivative)
22         # print("Error:" + str(error) + " Prediction:" + str(prediction))
23         up_prediction = input * (weight + step_amount)
24         up_error = (goal_prediction - up_prediction) ** 2
25         down_prediction = input * (weight - step_amount)
26         down_error = (goal_prediction - down_prediction) ** 2
27
28         if(down_error < up_error):
29             weight = weight - step_amount
30         if(down_error > up_error):
31             weight = weight + step_amount
32
33
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
Error:0.1196122250000007 Prediction:0.6541499999999999
Error:0.052526910156250044 Prediction:0.7708124999999999
Error:0.012661875624999997 Prediction:0.887475
Error:1.7118906250001052e-05 Prediction:1.0041375000000001
0.00413750000000127
PS C:\Projects\Honors2021\Second Semester\ITRI 626 AI\Assignment 1> c:: cd 'c:\Projects\Honors2021\Second Semester\ITRI 626 AI\Assignment 1'; & 'C:\Python39\python.exe' 'c:\Users\y\thon.python-2021.8.1105858891\pythonFiles\lib\python\debugpy\launcher' '62591' '--' 'c:\Projects\Honors2021\Second Semester\ITRI 626 AI\Assignment 1\Assignment 1.py'
-1.625
-1.6099999999999999
-1.595
-1.58
0.79
PS C:\Projects\Honors2021\Second Semester\ITRI 626 AI\Assignment 1> 
```

Figure 1: Getting the derivatives

According to (Mesquita, 2021), these four amounts showed me that I had to increase my step amount. I also made the `number_of_iterations` 25 at random.

This is based on a plot of a quadratic function.



Plot of a quadratic function

The error is given by the y-axis. If you're in point A and want to reduce the error toward 0, then you need to bring the x value down. On the other hand, if you're in point B and want to reduce the error, then you need to bring the x value up. To know which direction you should go to reduce the error, you'll use the **derivative**. A derivative [explains exactly how a pattern will change](#).

Figure 2: Plot of a quadratic function (Mesquita, 2021)

After making my step amount 0.1, I realized that after my 8<sup>th</sup> iterations it started giving me the same two values in repeat, as shown in the screenshot below, and from what I could understand from Steven Tartakovsky (2017) this means that the graph has reached minimal optimal point, thus the prediction and error begin a loop of repetition between two points. Thus, making my number\_of\_iterations 8.

```
1 import numpy as np
2
3 def hot_and_cold_learning(): # Do not modify the function name
4
5     step_amount = 0.1 # Choose a step_amount here between 1e-6 and 1e-1
6     number_of_iterations = 25 # Choose the number of iterations between 1 and 100000
7
8     ### Do not modify the code from here ###
9
10    weight = 0.25
11    input = 0.75
12    goal_prediction = 1.0
13
14    for iteration in range(number_of_iterations):
15
16        prediction = input * weight
17        error = (prediction - goal_prediction) ** 2
18        derivative = 2 * (prediction - goal_prediction)
19        print(derivative)
20        # print("Error:" + str(error) + " Prediction:" + str(prediction))
21        up_prediction = input * (weight + step_amount)
22        up_error = (goal_prediction - up_prediction) ** 2
23        down_prediction = input * (weight - step_amount)
24        down_error = (goal_prediction - down_prediction) ** 2
25
26        if(down_error < up_error):
27            weight = weight - step_amount
28        if(down_error > up_error):
29            weight = weight + step_amount
30
31    return abs(prediction - goal_prediction)
32
33    ### Do not modify the code to here ###
34
35 print(hot_and_cold_learning())
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
0.0250000000000000355
-0.125
0.0250000000000000355
-0.125
0.0250000000000000355
-0.125
0.0250000000000000355
-0.125
0.0250000000000000355
-0.125
0.0250000000000000355
-0.125
0.0625
PS C:\Projects\Honors2021\Second Semester\ITRI 626 AI\Assignment 1> |
```

Figure 3: Getting repetitions of predictions

After changing the number\_of\_iterations to 8 I started playing around with the step\_amount, while having the plot of a quadratic function in my head. I also tried using numpy for the graphs but felt more comfortable drawing it out.

My drawings on a white board is shown in the screenshot below.

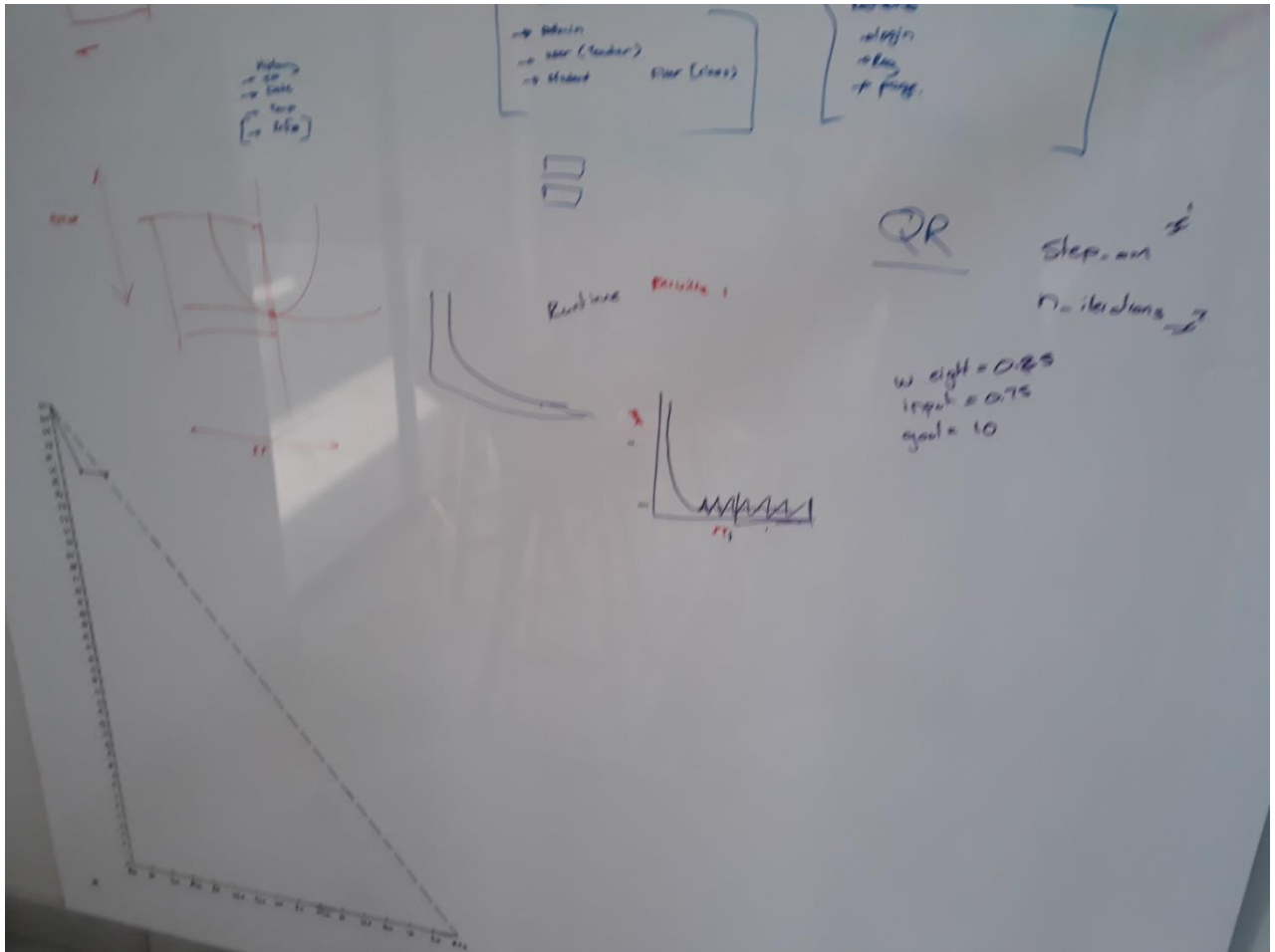


Figure 4: Drawings on board

## Outcome

After playing around I figured out that the best outcome was a step\_amount of 0.15555 and a number\_of\_iterations of 8.

This gave me an error of 1.7118906250001052e-05 and a prediction of 1.0041375000000001 with a result of 0.004137500000000127. As show in the screenshot below.

```
Assignment 1.py U X
C: > Projects > Honors2021 > Second Semester > ITRI 626 AI > Assignment 1 > Assignment 1.py > hot_and_cold_learning
1 import numpy as np
2
3 def hot_and_cold_learning(): # Do not modify the function name
4
5     step_amount = 0.15555 # Choose a step_amount here between 1e-6 and 1e-1
6     number_of_iterations = 8 # Choose the number of iterations between 1 and 100000
7
8     ### Do not modify the code from here ###
9
10    weight = 0.25
11    input = 0.75
12    goal_prediction = 1.0
13
14    for iteration in range(number_of_iterations):
15
16        prediction = input * weight
17        error = (prediction - goal_prediction) ** 2
18        # derivative = 2 * (prediction - goal_prediction)
19        # print(derivative)
20        print("Error:" + str(error) + " Prediction:" + str(prediction))
21        up_prediction = input * (weight + step_amount)
22        up_error = (goal_prediction - up_prediction) ** 2
23        down_prediction = input * (weight - step_amount)
24        down_error = (goal_prediction - down_prediction) ** 2
25
26        if(down_error < up_error):
27            weight = weight - step_amount
28        if(down_error > up_error):
29            weight = weight + step_amount
30
31    return abs(prediction - goal_prediction)
32
33    ### Do not modify the code to here ###
34
35    print(hot_and_cold_learning())

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
python.python-2021.8.1105858891\pythonFiles\lib\python\debugpy\launcher '58034' '--' 'c:\Projects\Honors2021\Second Semester\ITRI 626 AI\Assignment 1\Assignment 1.py'
Error:0.66015625 Prediction:0.1875
Error:0.48418982640625 Prediction:0.3041625
Error:0.335443680625 Prediction:0.42082499999999995
Error:0.2139178126562501 Prediction:0.53748749999999999
Error:0.11961222250000007 Prediction:0.65414999999999999
Error:0.052526910156250044 Prediction:0.77081249999999999
Error:0.01266187562499997 Prediction:0.887475
Error:1.7118906250001052e-05 Prediction:1.0041375000000001
0.00413750000000127
PS C:\Projects\Honors2021\Second Semester\ITRI 626 AI\Assignment 1> 
```

Figure 5: Final outcome

## References

Mesquita, D. (2021). *Python AI: How to Build a Neural Network & Make Predictions*.

<https://realpython.com/python-ai-neural-network/>

Steven Tartakovsky, S. C. a. M. M. (2017). *Deep Learning Hyperparameter Optimization with Competing Objectives*. <https://developer.nvidia.com/blog/sigopt-deep-learning-hyperparameter-optimization/>